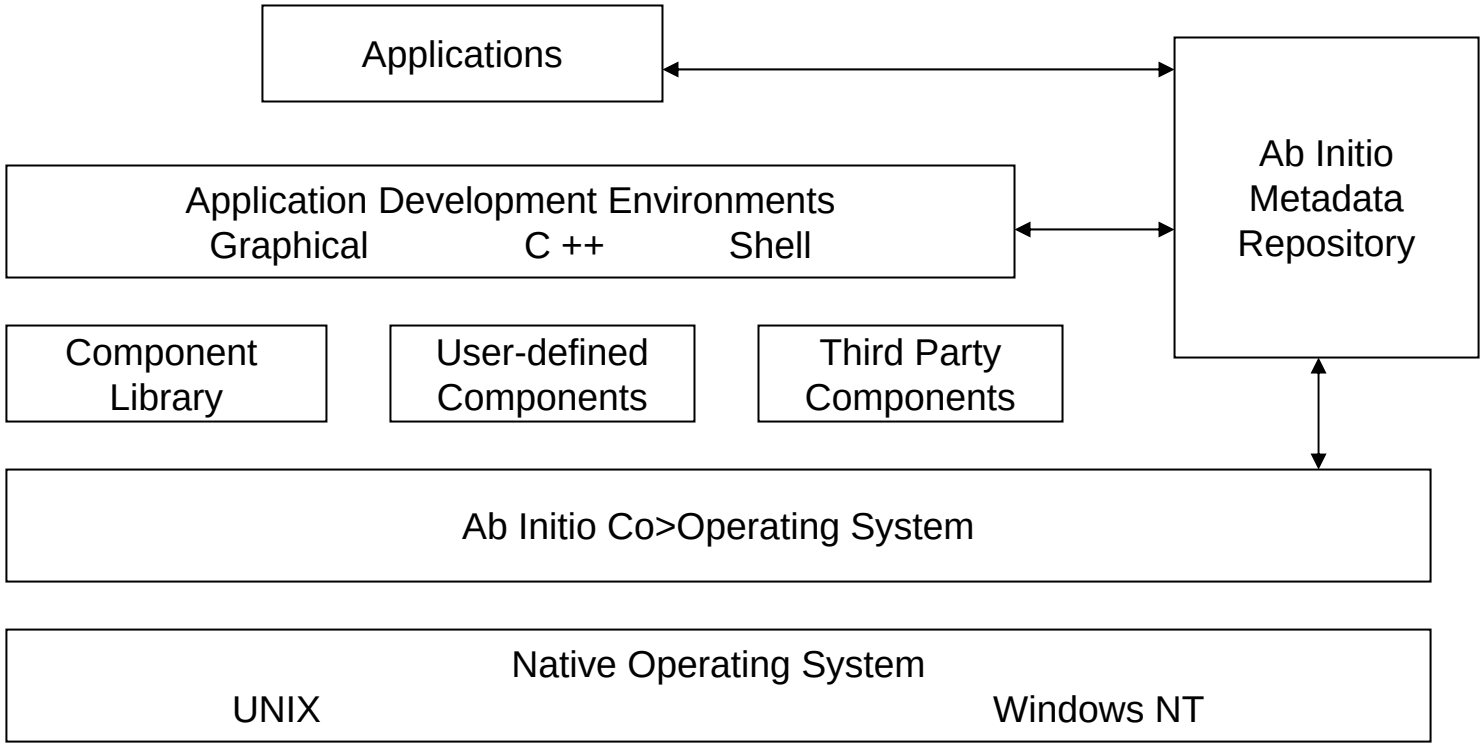
Ab Initio Architecture

By:
Arun Ravindranath
172055
L1/L2 Application Support

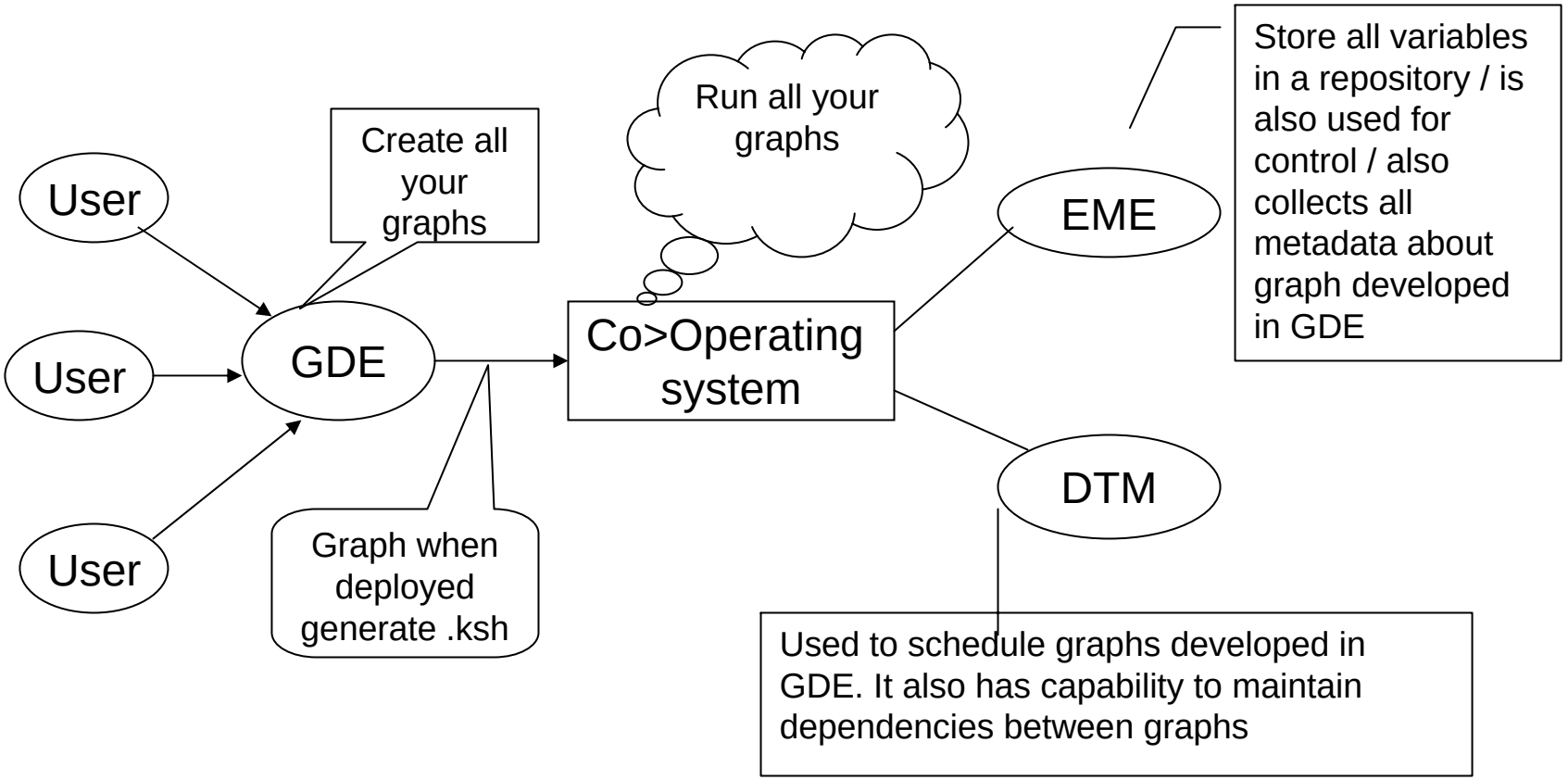
About Ab Initio

- Ab Initio is a general purpose data processing platform for enterprise class, mission critical applications such as data warehousing, clickstream processing, data movement, data transformation and analytics.
- Supports integration of arbitrary data sources and programs, and provides complete metadata management across the enterprise.
- Proven best of breed ETL solution.
- Applications of Ab Initio:
  - ETL for data warehouses, data marts and operational data sources.
  - Parallel data cleansing and validation.
  - Parallel data transformation and filtering.
  - High performance analytics
  - Real time, parallel data capture.

Ab Initio Architecture



Ab Initio Overview



Co>Operating System

- The Co>Operating System is core software that unites a network of computing resources-CPUs, storage disks, programs, datasets-into a production-quality data processing system with scalable performance and mainframe reliability.

- The Co>Operating System is layered on top of the native operating systems of a collection of computers. It provides a distributed model for process execution, file management, process monitoring, check-pointing, and debugging.

Co>Operating System

- The Graphical Development Environment (GDE) provides a graphical user interface into the services of the Co>Operating System.
- Unlimited scalability : Data parallelism results in speedups proportional to the hardware resources provided, double the number of CPUs and execution time is halved.
- Flexibility : Provides a powerful and efficient data transformation engine and an open component model for extending and customizing Ab Initio's functionality.
- Portability : Runs heterogeneously across a huge variety of operating system and hardware platforms.

Graphical Development Environment (GDE)

- GDE lets create applications by dragging and dropping components onto a canvas configuring them with familiar, intuitive point and click operations, and connecting them into executable flowcharts.

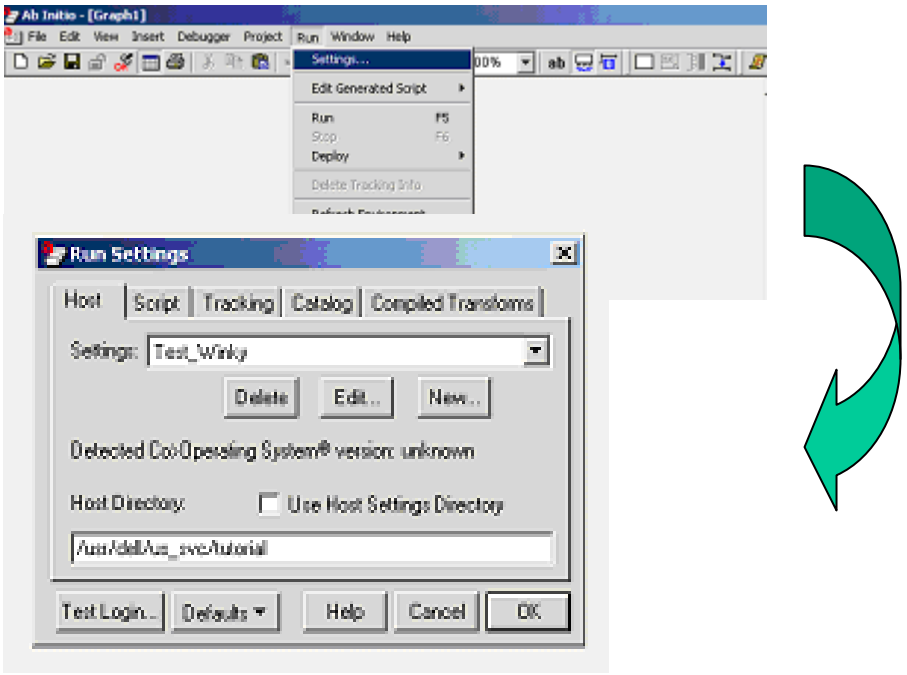
- These diagrams are architectural documents that developers and managers alike can understand and use, but they are not mere pictures: the co>operating system executes these flowcharts directly. This means that there is a seamless and solid connection between the abstract picture of the application and the concrete reality of its execution.

Ab Initio S/w Versions & File Extensions

- Software Versions
  - Co>Operating System Version =>
  - GDE Version =>

- File Extensions
  - .mp Stored Ab Initio graph or graph component
  - .mpc Program or custom component
  - .mdc Dataset or custom dataset component
  - .dml Data Manipulation Language file or record type definition
  - .xfr Transform function file
  - .dat Data file (either serial file or multifile)

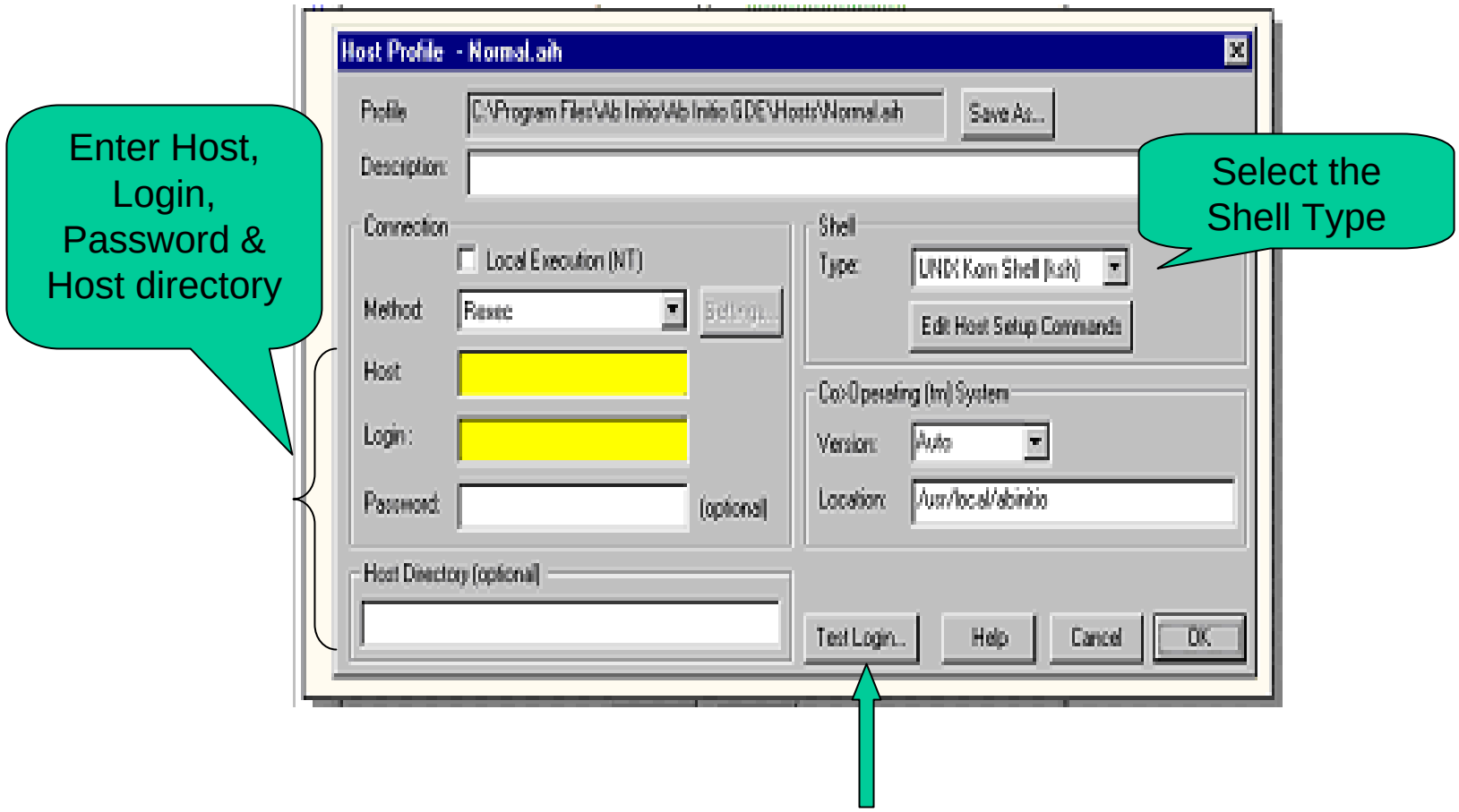
Connecting to Co>op Server from GDE



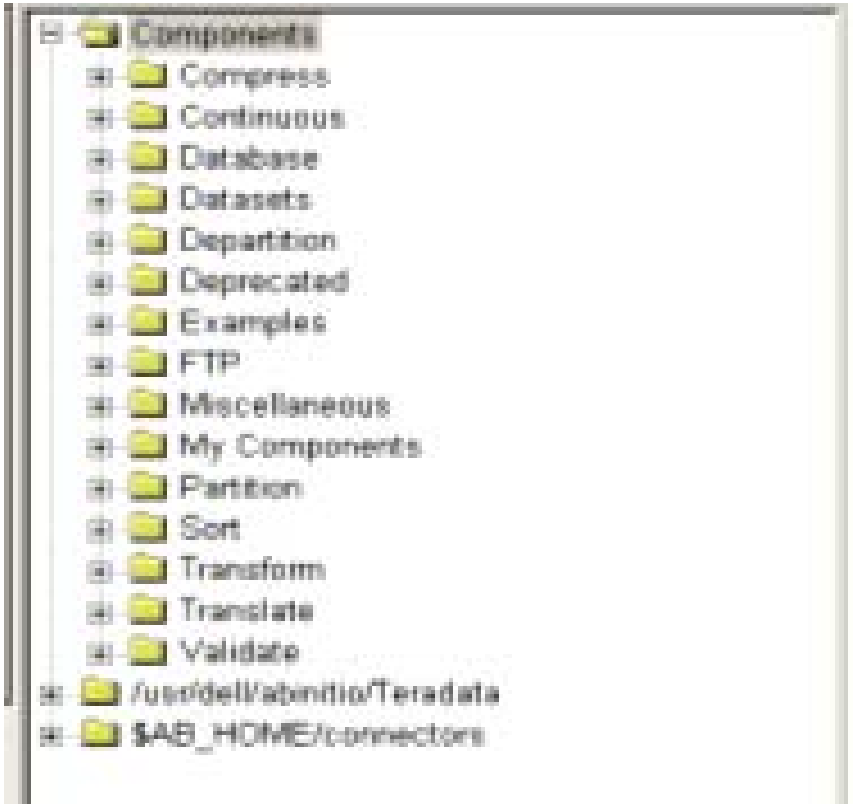
Host Profile Setting

1. Choose settings from the run menu
2. Check the use host profile setting checkbox.
3. Click Edit button to open the Host profile dialog.
4. If running Ab Initio on your local NT system, check Local Execution (NT) checkbox and go to step 6.
5. If running Ab Initio on a Remote UNIX system, fill in the path to the Host and Host Login and Password.
6. Type the full path of Host directory.
7. Select the Shell Type from pull down menu.
8. Test Login and if necessary make changes.

Host Profile

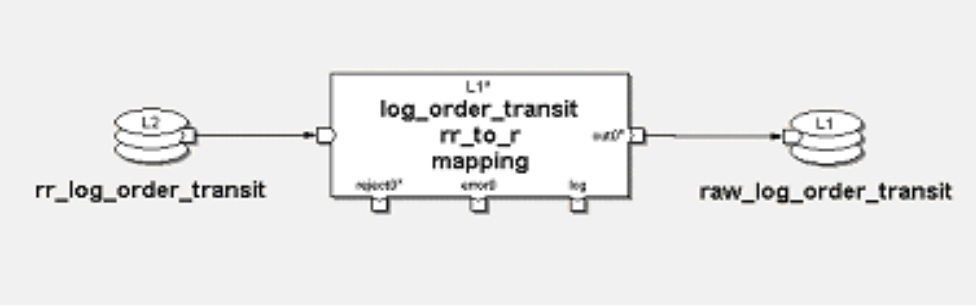


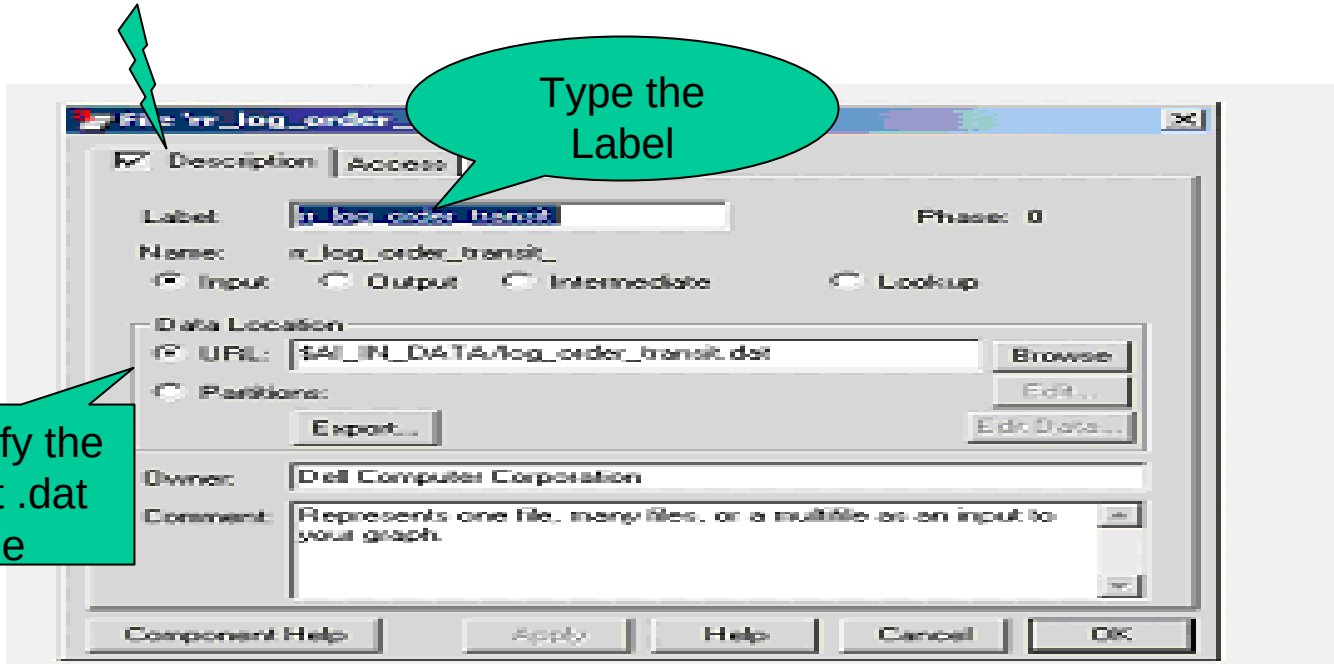
Ab Initio Components



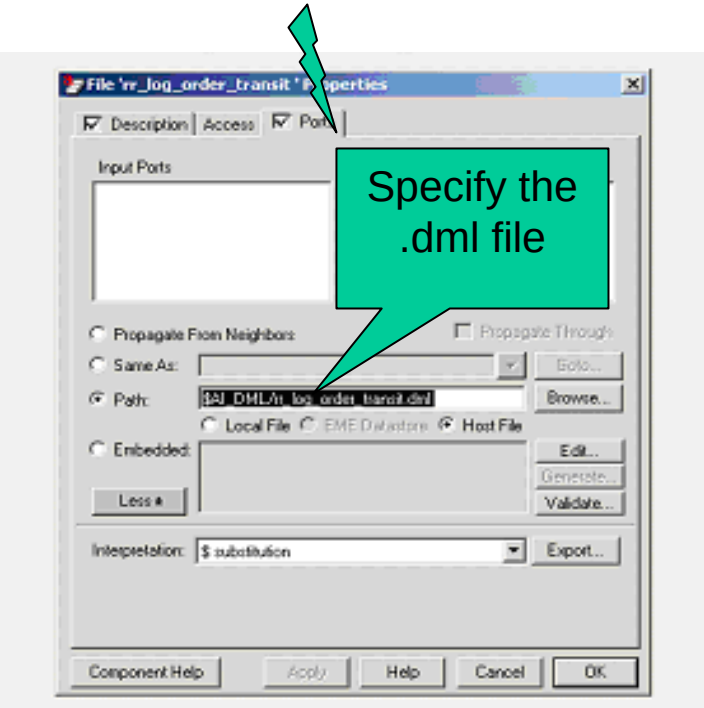
Ab Initio provided components. Datasets, Partition, Transform, Sort, Database are frequently used.

Creating Graph



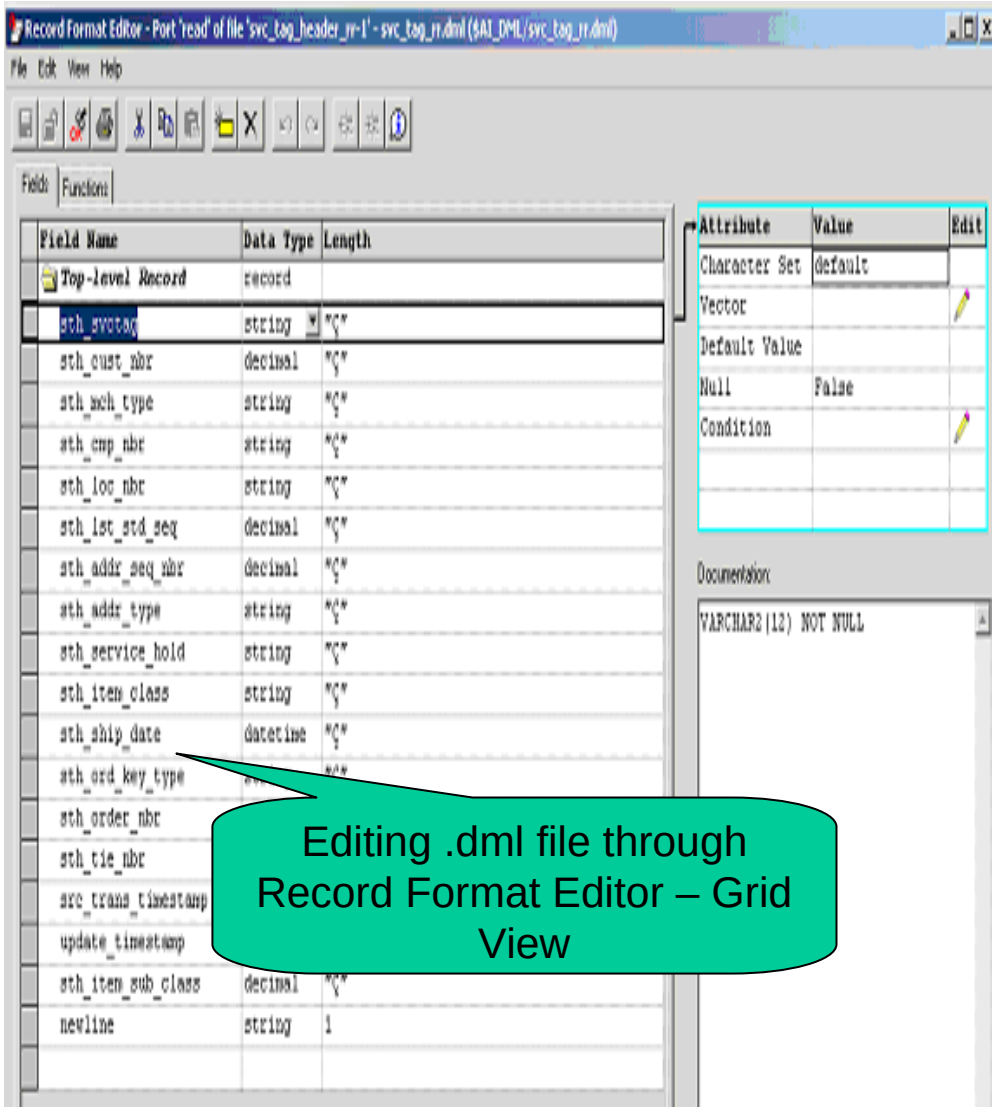


Create Graph - Dml



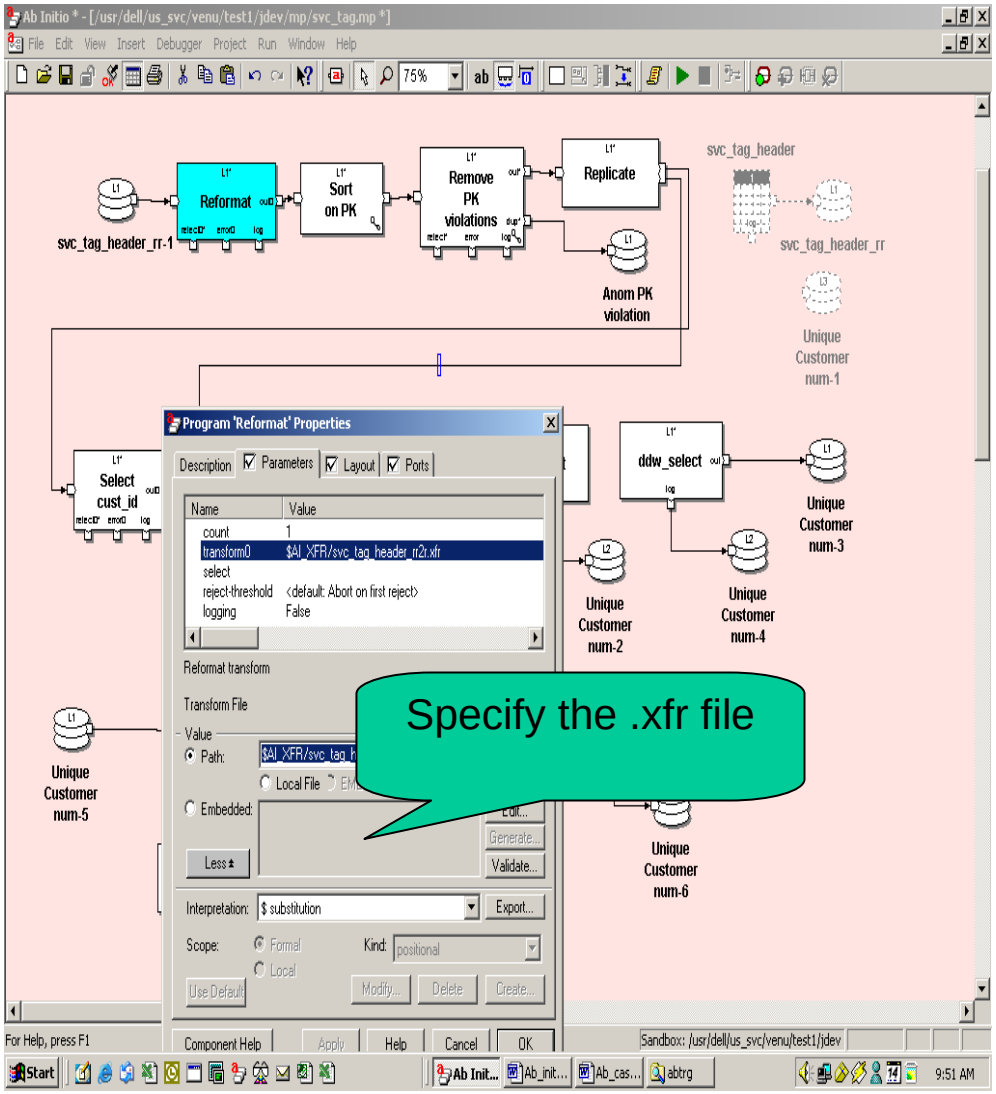
- Propagate from Neighbors: Copy record formats from connected flow.
- Same As: Copy record format's from a specific component's port.
- Path: Store record formats in a Local file, Host File, or in the Ab Initio repository.
- Embedded: Type the record format directly in a string.

Creating Graph - dml



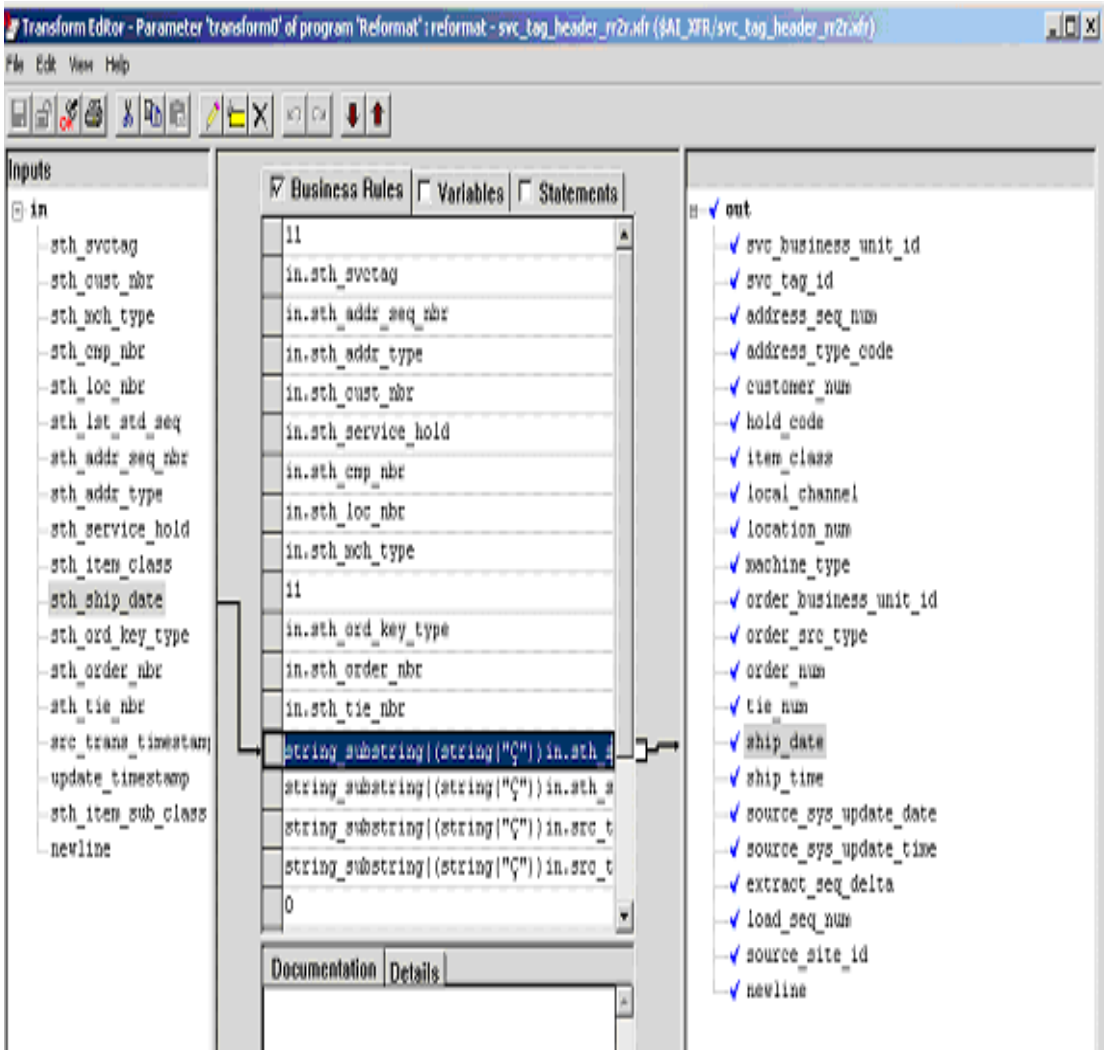
- DML is Ab Initio's Data Manipulation Language.
- DML describes data in terms of
  - Record Formats that list the fields and format of input, output, and intermediate records.
  - Expressions that define simple computations, for example, selection.
  - Transform Functions that control reformatting, aggregation, and other data transformations.
  - Keys that specify groupings, ordering, and partitioning relationships between records.

Creating Graph - Transform



- A transform function is either a DML file or a DML string that describes how you manipulate your data.
- Ab Initio transform functions mainly consist of a series of assignment statements. Each statement is called a business rule.
- When Ab Initio evaluates a transform function, it performs following tasks:
  - Initializes local variables
  - Evaluates statements
  - Evaluates rules.
- Transform function files have the xfr extension.

Creating Graph - xfr

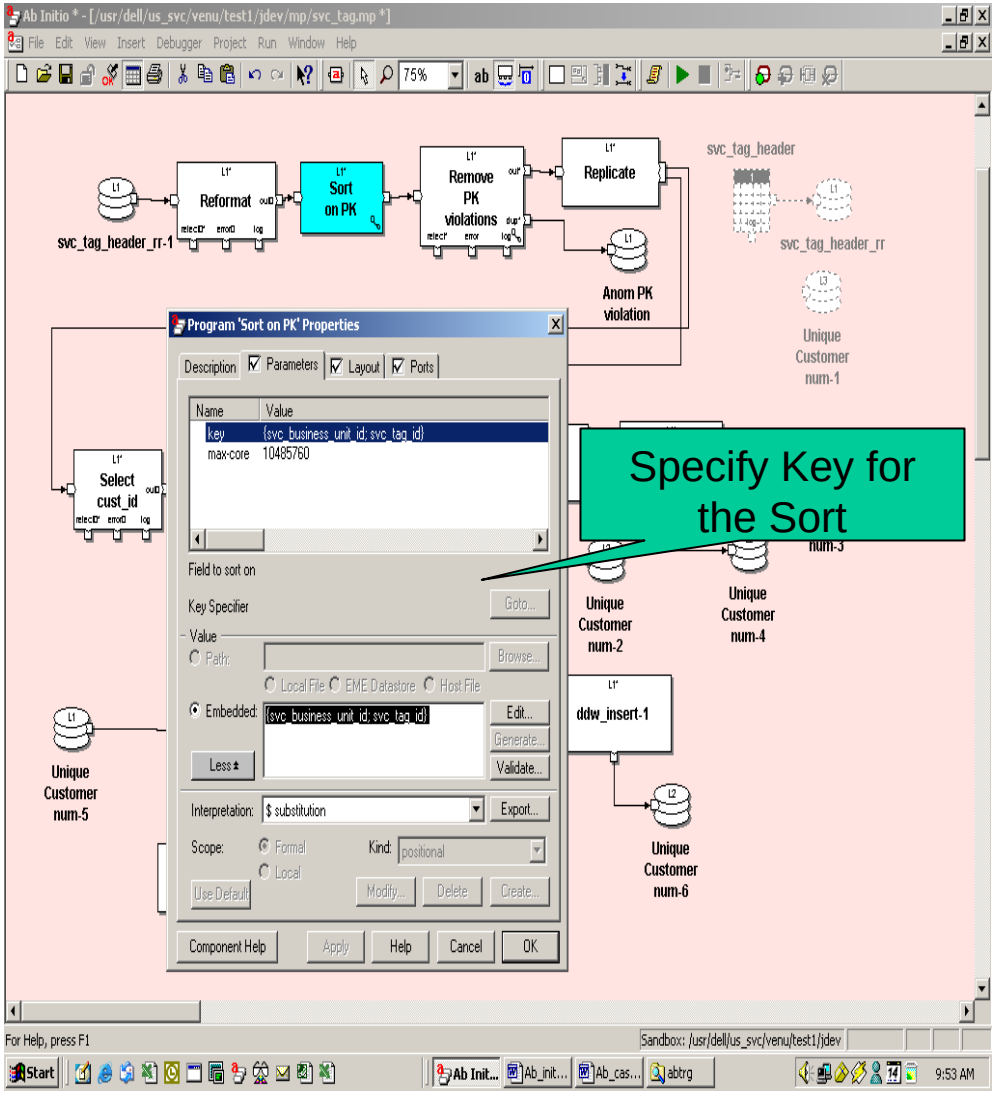


- Transform functions: A set of rules that compute output values from input values.
- Business rule: Part of a transform function that describes how you manipulate one field of your output data.
- Variable: Optional part of a transform function that provides storage for temporary values.
- Statement: Optional part of a transform function that assigns values of variables in a specific order.

Sample Components

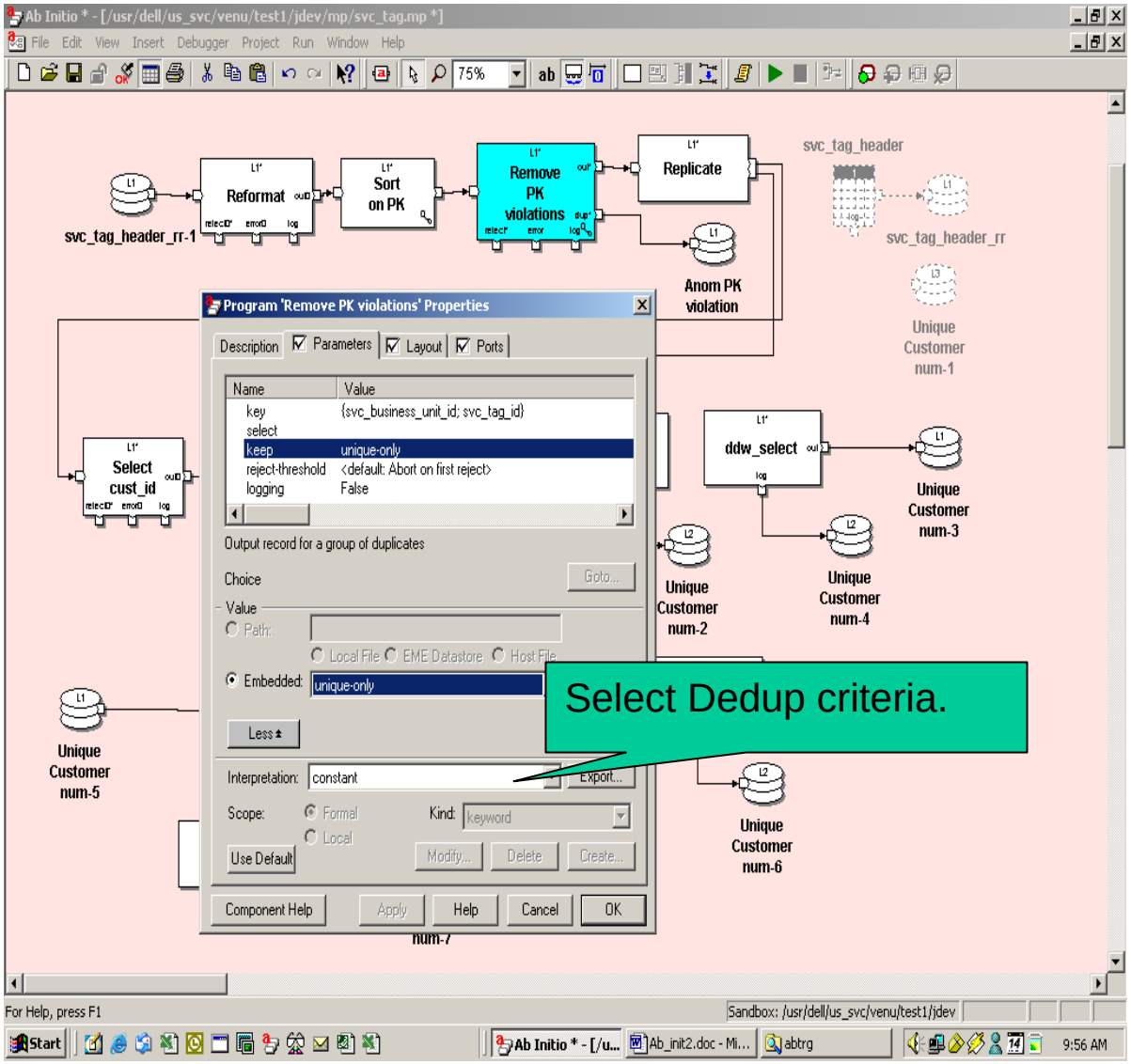
- Sort
- Dedup
- Join
- Replicate
- Rollup
- Filter by Expression
- Merge
- Lookup
- Reformat etc.

Creating Graph – Sort Component



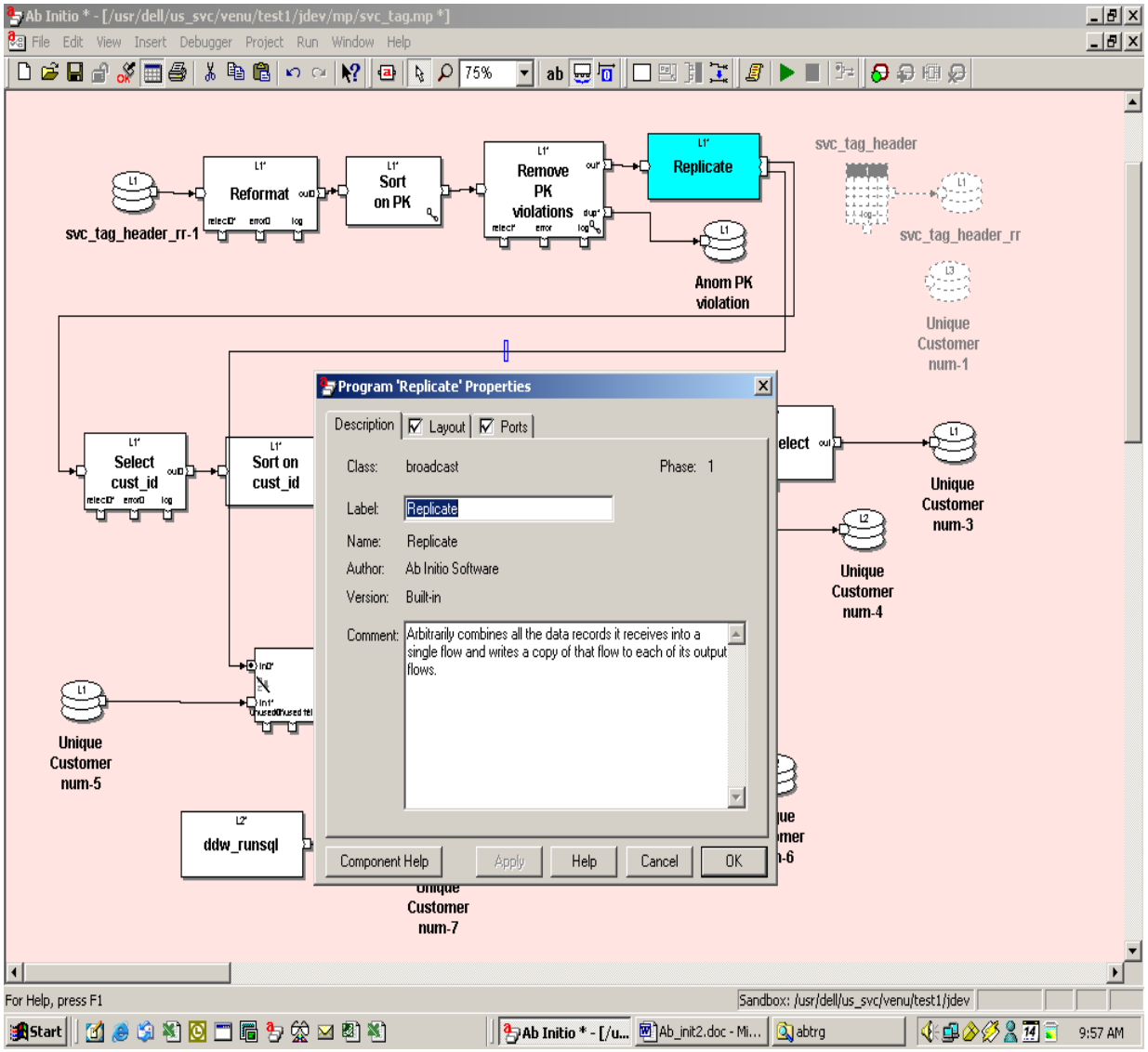
- Sort: The sort component reorders data. It comprises two parameters: Key and max-core.
- Key: The Key is one of the parameters for Sort component which describes the collation order.
- Max-core: The max-core parameter controls how often the sort component dumps data from memory to disk.

Creating Graph – Dedup component



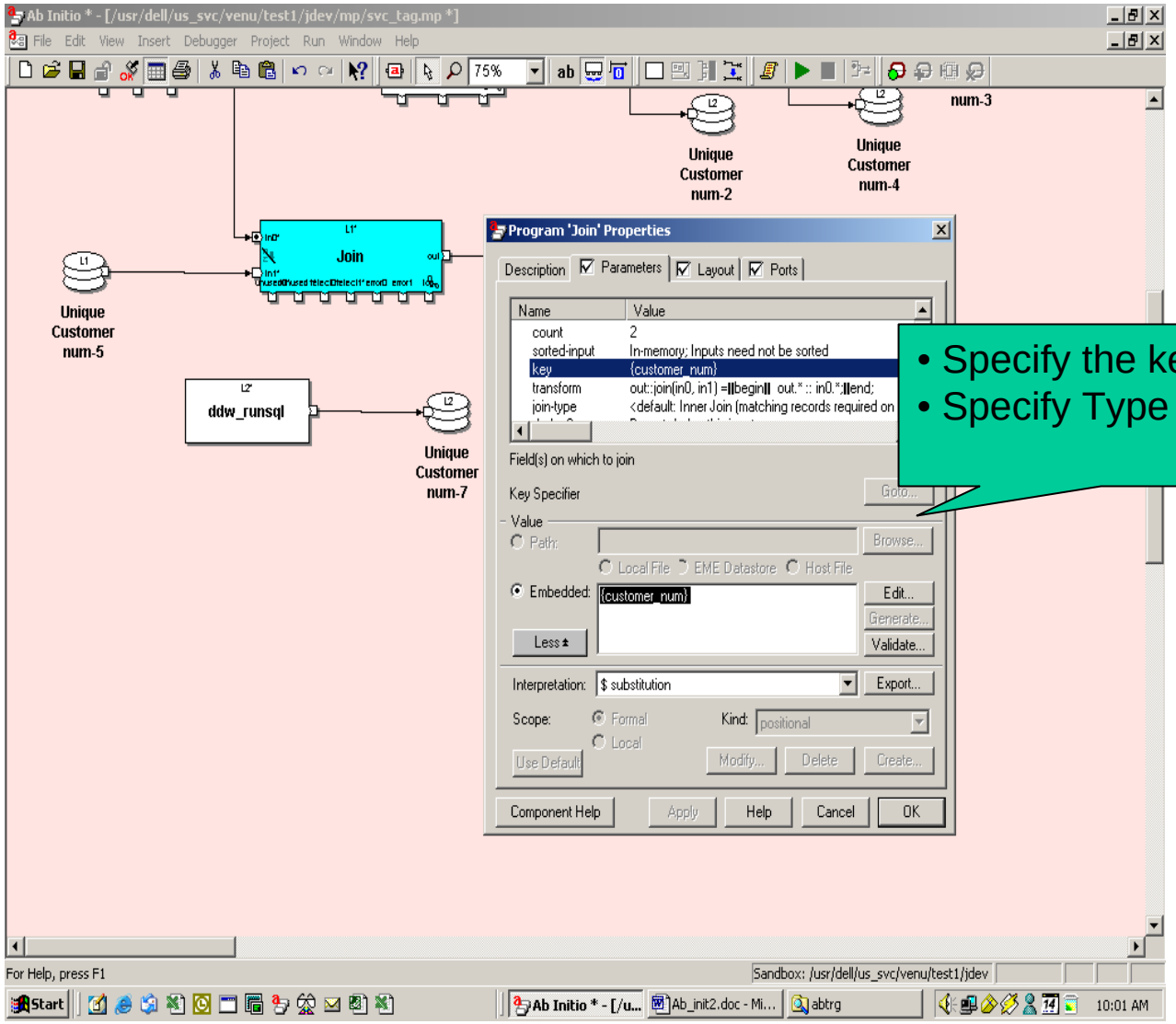
- Dedup component removes duplicate records.
- Dedup criteria will be either unique-only, First or Last.

Creating Graph – Replicate Component



- Replicate combines the data records from the inputs into one flow and writes a copy of that flow to each of its output ports.
- Use Replicate to support component parallelism.

Creating Graph – Join Component



Database Configuration (.dbc)

- A file with a .dbc extension which provides the GDE with the information it needs to connect to a database. A configuration file contains the following information:
  - The name and version number of the database to which you want to connect.
  - The name of the computer on which the database instance or server to which you want to connect runs, or on which the database remote access software is installed.
  - The name of the database instance, server, or provider to which you want to connect.
  - You generate a configuration file by using the Properties dialog box for one of the Database components.

- Types of Parallel Processing
  - Component-level Parallelism: An application with multiple components running simultaneously on separate data uses component parallelism.
  - Pipeline parallelism: An application with multiple components running simultaneously on the same data uses pipeline parallelism.
  - Data Parallelism: An application with data divided into segments that operates on each segment simultaneously uses data parallelism.

Partition Components

- Partition by Expression: Dividing data according to a DML expression.
- Partition by Key: Grouping data by a key.
- Partition with Load balance: Dynamic load balancing.
- Partition by Percentage: Distributing data, so the output is proportional to fractions of 100.
- Partition by Range: Dividing data evenly among nodes, based on a key and a set of partitioning ranges.
- Partition by Round-robin: Distributing data evenly, in blocksize chunks, across the output partitions.

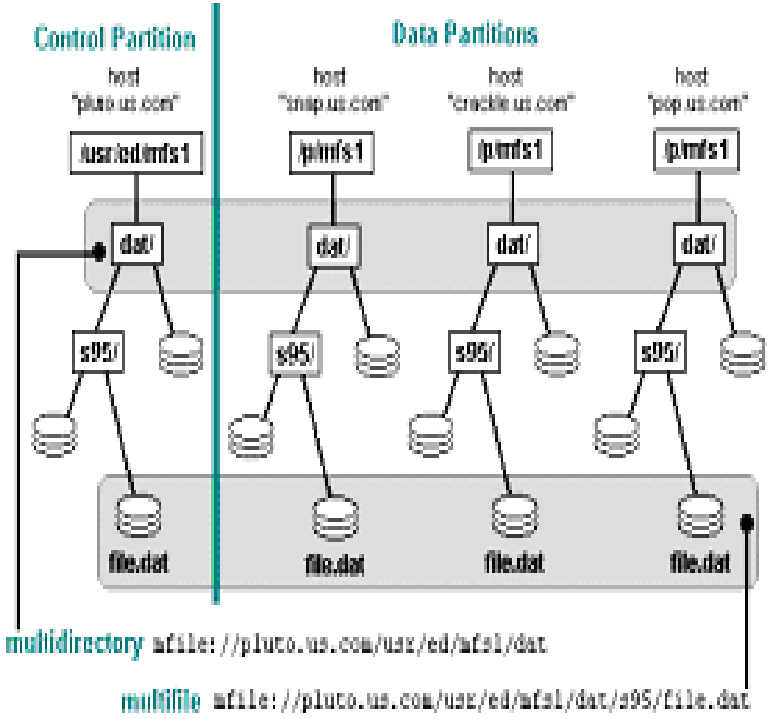
Departition Components

- Concatenate: Concatenate component produces a single output flow that contains first all the records from the first input partition, then all the records from the second input partition and so on.
- Gather: Gather component collects inputs from multiple partitions in an arbitrary manner, and produces a single output flow, does not maintain sort order.
- Interleave: Interleave component collects records from many sources in round robin fashion.
- Merge: Merge component collects inputs from multiple sorted partitions and maintains the sort order.

Multifile systems

- A multifile system is a specially created set of directories, possibly on different machines, which have identical substructure.
- Each directory is a partition of the multifile system. When a multifile is placed in a multifile system, its partitions are files within each of the partitions of the multifile system.
- Multifile system leads to better performance than flat file systems because multifile systems can divide your data among multiple disks or CPUs.
- Typically (SMP machine is exception) a multifile system is created with the control partition on one node and data partitions on other nodes to distribute the work and improve performance.
- To do this use full internet URLs that specify file and directory names and locations on remote machines.

Multifile


Control Partition
Data Partitions
host
"pluto.us.com"
host
"snip.us.com"
host
"crackle.us.com"
host
"pop.us.com"
/usr/ed/mfs1
/p/mfs1
/p/mfs1
/p/mfs1
dat/
dat/
dat/
dat/
s95/
s95/
s95/
s95/
file.dat
file.dat
file.dat
file.dat
multidirectory mfile://pluto.us.com/usr/ed/mfs1/dat
multifile mfile://pluto.us.com/usr/ed/mfs1/dat/s95/file.dat

SANDBOX

- A sandbox is a collection of graphs and related files that are stored in a single directory tree, and treated as a group for purposes of version control, navigation, and migration.
- A sandbox can be a file system copy of a datastore project.

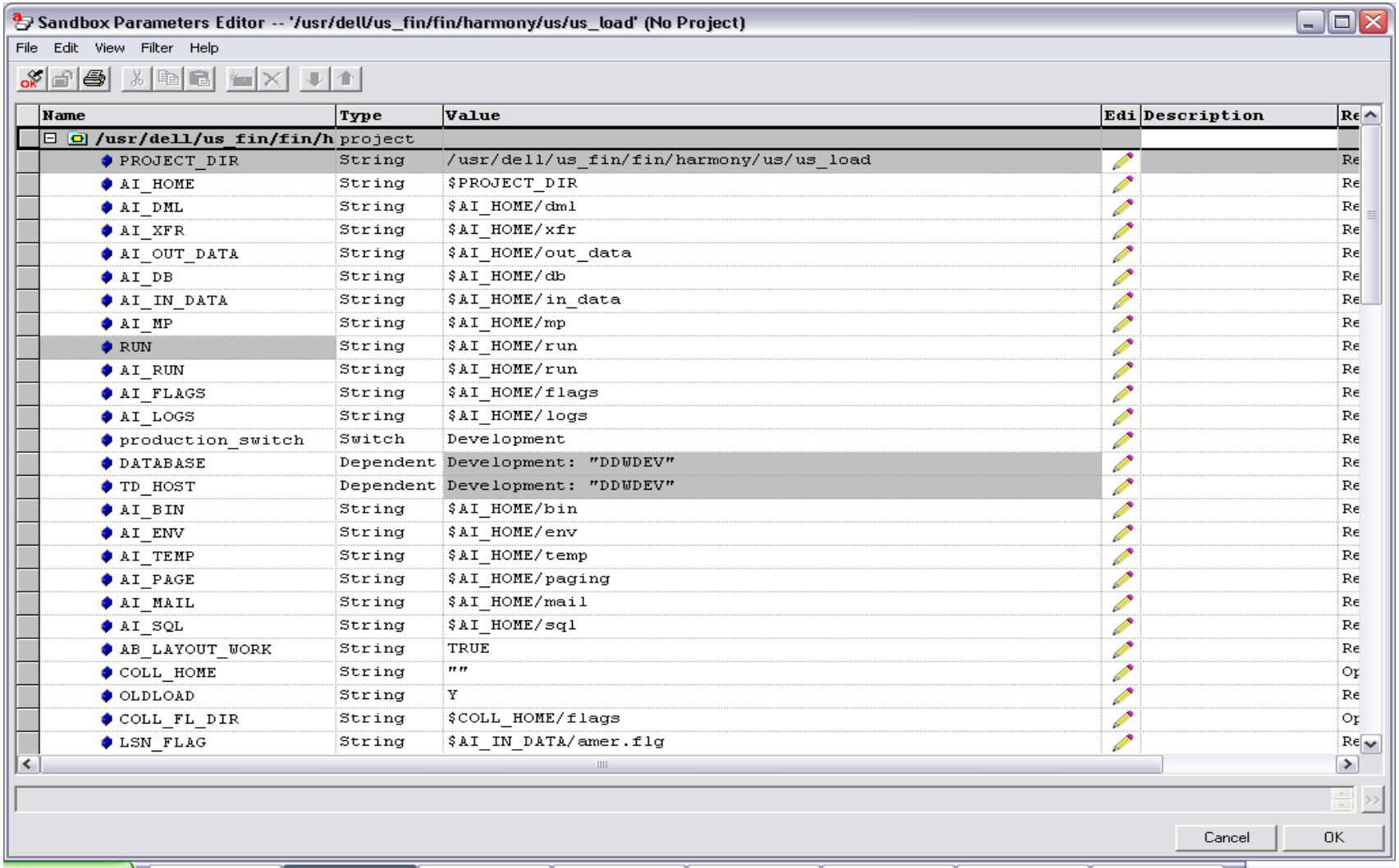
- In the graph, instead of specifying the entire path for any file location ,we specify only the sandbox parameter variable. For ex : $AI_IN_DATA/customer_info.dat. where $AI_IN_DATA contains the entire path with reference to the sandbox $AI_HOME variable.

- The actual in_data dir is $AI_HOME/in_data in sandbox

SANDBOX

- The sandbox provides an excellent mechanism to maintain uniqueness while moving from development to production environment by means switch parameters.
- We can define parameters in sandbox those can be used across all the graphs pertaining to that sandbox.
- The topmost variable $PROJECT_DIR contains the path of the home directory

SANDBOX


Sandbox Parameters Editor -- '/usr/dell/us_fin/fin/harmony/us/us_load' (No Project)
File Edit View Filter Help
Name Type Value Edi Description Re
/usr/dell/us_fin/fin/h project
PROJECT_DIR String /usr/dell/us_fin/fin/harmony/us/us_load
AI_HOME String $PROJECT_DIR
AI_DML String $AI_HOME/dml
AI_XFR String $AI_HOME/xfr
AI_OUT_DATA String $AI_HOME/out_data
AI_DB String $AI_HOME/db
AI_IN_DATA String $AI_HOME/in_data
AI_MP String $AI_HOME/mp
RUN String $AI_HOME/run
AI_RUN String $AI_HOME/run
AI_FLAGS String $AI_HOME/flags
AI_LOGS String $AI_HOME/logs
production_switch Switch Development
DATABASE Dependent Development: "DDWDEV"
TD_HOST Dependent Development: "DDWDEV"
AI_BIN String $AI_HOME/bin
AI_ENV String $AI_HOME/env
AI_TEMP String $AI_HOME/temp
AI_PAGE String $AI_HOME/paging
AI_MAIL String $AI_HOME/mail
AI_SQL String $AI_HOME/sql
AB_LAYOUT_WORK String TRUE
COLL_HOME String ""
OLDLOAD String Y
COLL_FL_DIR String $COLL_HOME/flags
LSN_FLAG String $AI_IN_DATA/amer.flg
Cancel OK
start Querym... Ab Initio ... GDE Help 2 Note... Start Menu Ab Initio ... Sent Ite... harp.us.... 10:50 AM

Deploying

- Every graph after validation and testing has to be deployed as .ksh file into the run directory on UNIX.
- This .ksh file is an executable file which is the backbone for the entire automation/wrapper process.
- The wrapper automation consists of .run, .env, dependency list ,job list etc
- For a detailed description on wrapper and different directories and files , Please refer the documentation on wrapper / UNIX presentation.

References

- Ab Initio Tutorial
- Ab Initio Online Help
- Website (abinitio.com)